Universidade do Porto

FEUP Faculdade de Engenharia

MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES, RAMO: TELECOMUNICAÇÕES, ELETRÓNICA E COMPUTADORES

DEVICE DRIVER

Unidade Curricular de Sistemas Operativos

Professor Pedro Alexandre Guimarães Lobo Ferreira Souto
Professor José Manuel de Magalhães Cruz

André Duarte Correia de Oliveira, 201405639
Baltasar de Vasconcelos Dias Aroso, 201404125

Maio 2017

Desenvolvimento do DD

Completámos a Lab3 (DD com polling) tendo inclusive implementado algumas melhorias como redução de variáveis globais. Inclui a leitura/escrita de strings e ficheiros.

Concluímos também a Lab4 (DD com interrupts), tendo neste caso concluído os enhancements com exceção de adição de funcionalidades FIFO. O nosso DD permite escritas com tamanho superior ao do buffer (KFIFO), mas leituras com tamanho igual ou inferior, como permitido nas especificações.

Enhancements

2. Error handling:

Os devidos erros foram detetados e são retornados os valores correspondentes a cada situação de erro.

3.1. Redução de variáveis globais:

Apenas recorremos a variáveis globais na função init() e exit(), recorrendo, por exemplo, à função container_of() (ver função open()).

3.2. Eliminação de race conditions:

Eliminámos as race conditions com spinlocks, kfifo (na recolha ou transmissão de dados) e simple sleeping.

3.3. Implementação do funcionamento da flag O_NONBLOCK:

Caso o utilizador indique a flag O_NONBLOCK aquando da abertura do ficheiro, a função read/write deve retornar imediatamente. Read: caso não haja dados no buffer; Write: caso o buffer esteja cheio. Testámos a flag juntamente com estas condições.

3.4. Adição de operações ioctl:

A função ioctl implementada permite alterar o número de bits, número de stop bits, paridade e bitrate.
O ficheiro de teste ioctl permite ao utilizador escolher os valores desejados dentro da gama de valores possível, fazendo as devidas mudanças no DD e, no fim, comparando os valores do DD com os especificados.

3.5. Interrupção da função read:

Quando o utilizador decide terminar o processo, a função wait_event_interruptible_timeout() retorna -ERESTARTSYS. Uma simples verificação desse valor de retorno permite interrompter o processo de leitura/escrita.

3.6. Limite do número de users com acesso à porta série:

O incremento de uma variável global “users” em open() e decremento em release(), protegida por spinlocks, permite controlar o acesso de utilizadores à porta série através da avaliação do seu valor.

3.7. Adição de funcionalidades FIFO:

Inicializámos o FIFO Control Register (fcr) com os bits desejados.

3.8. Adição da funcionalidade poll/select:

Através da verificação do estado do buffer criámos a mask (de bits) de retorno.
Caso kfifo tenha length superior a 0, há algo a ler.
Caso kfifo tenha length inferior a SIZE_KFIFO (capacidade deste), há possibilidade de escrita.

Testes

Para cada teste, há uma descrição ao longo do código que explica o seu funcionamento. Para além disso, para o poll e O_NONBLOCK há dois ficheiros *.txt que devem ser lidos antes do seu teste.